Stock-out Stopper

Joseph Salmento & Elliott Clark

Product Concept/Overview

- Stock-out Stopper is an inventory demand forecasting application.
- Predicts daily, weekly, or monthly sales on a per item basis
- Explains how it arrived at its sales prediction to increase retailer confidence in application
- Allows retailers to ensure adequate inventory of in demand items and minimize inventory overhead

Product AI Canvas

Opportunity:
-Retailers can optimize costs with more precise sales predictions and avoid loss of potential revenue from unforeseen stock outs

Consumers:
-Retailers (brick and mortar and e-commerce)
-Also extendible to any system that requires an inventory (i.e. maintenance parts)

Strategy:
-Simple model for explainability
-Easy to use and provide sales data

Solution:
-Sales predictions are generated from a Decision Tree Regressor trained on available historical data for the store

Data:
-Historical Sales Data from Kaggle

Success Criteria:
-Retailer using app to decide inventory levels
-Retailers providing sales data through app

Product Team

Joe Salmento:

- Data Engineer/ Data Scientist

- Built data pipeline

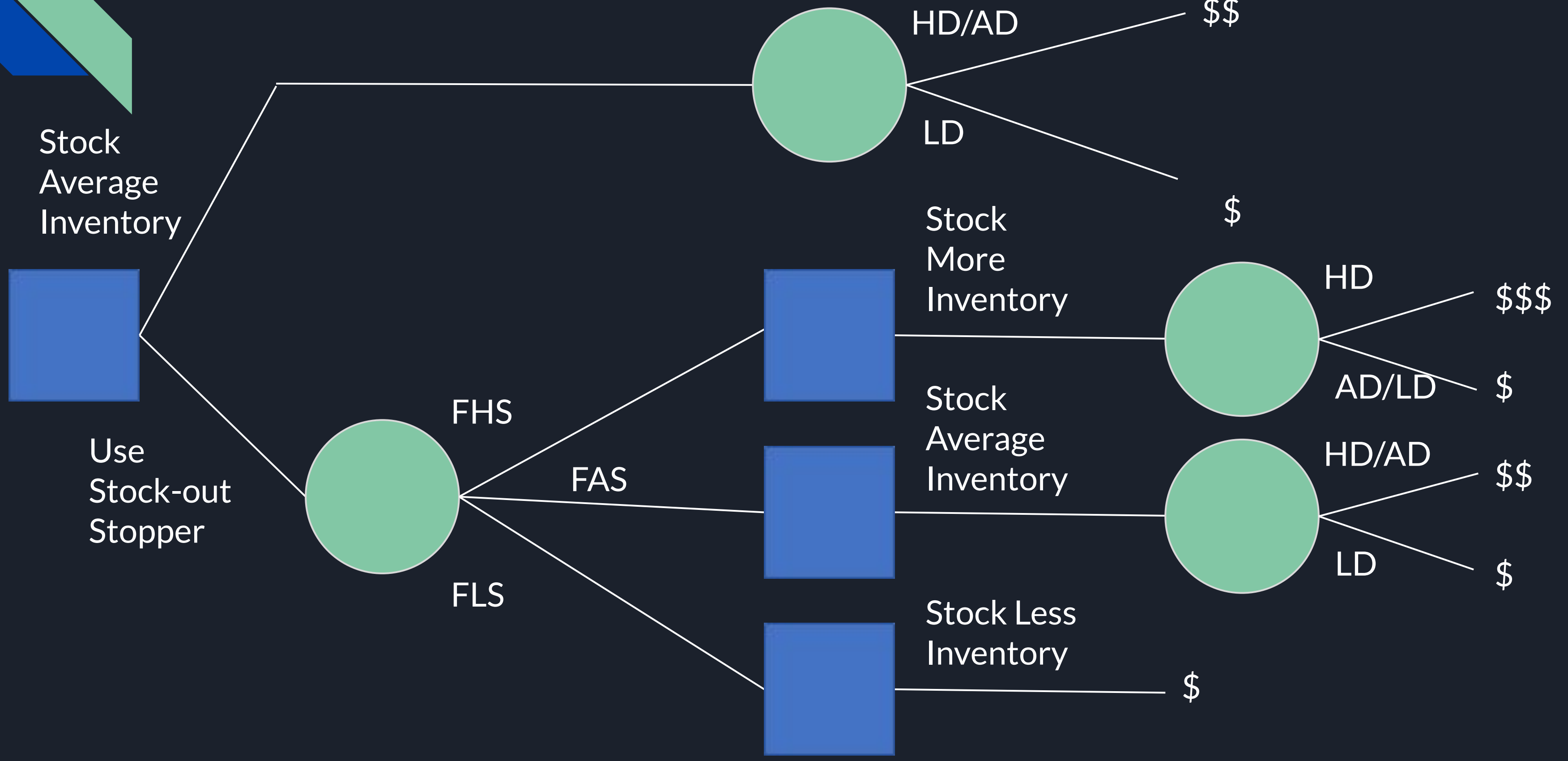
- Built sales forecasting model

Elliott Clark:

- Software Engineer

- Built MVP app

Value of Data Calculation for Retailer



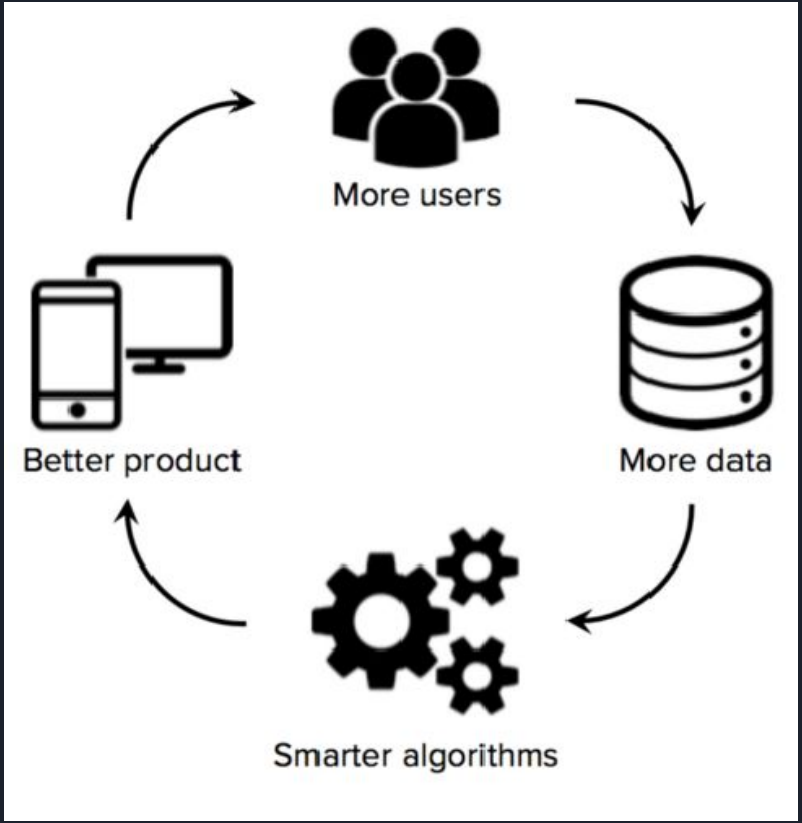
Data flywheel / Data Network effects

Retailers use Stock-out Stopper to predict inventory and make money

Retailers provide use with additional sales data.

Use additional sales data to produce better/customized sales models

Better sales models result in retailers making more money and more retailers using Stock-out Stopper



Model type selection, metrics

Decision Tree Regressor

- Targeting small business without their own forecasting software
- Inventory Decisions need to be explainable.
- Decision Trees have the best explainability.

Metrics:

- Primary metric R^2
- Other metrics looked at MSE, MAE, and Max Error
- # Over predictions, #Under predictions

Proposed Architecture

Data Engineering

Data Versioning: DVC

Data Exploration: Python Libraries (Pandas, Matplotlib, Seaborn)

Data Warehouse/Lake: AWS S3

Data Pipelines: Airflow

Data Science

Model Versioning: MLFlow

Modeling Framework: Sklearn

Local Development Environments: JupyterLab, VSCode

Infrastructure Management: Docker

Software Engineering

MVP Demo: Streamlit

CI/CD and Testing: Github

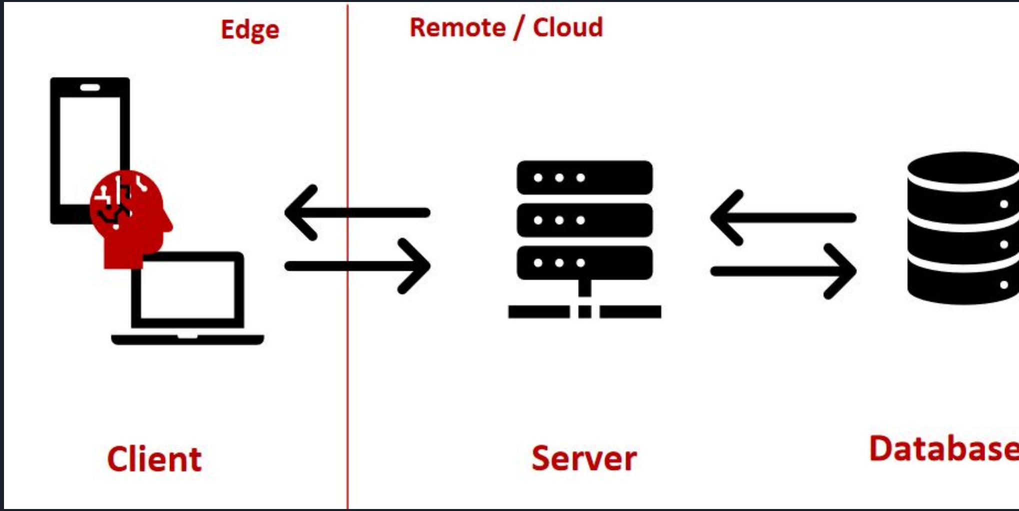
Deployment Orchestration: Docker

Model Deployment and Monitoring

Deploy application to the edge. Simple model so not worried about hardware limitations. Retailers will always have access to make inventory decisions

Montor model monthly to avoid drift using evidently.ai

Test model on each newly provided dataset from retailer



Ethical Considerations

Ethical Considerations:

- Ensure Retailers are aware of how their data may be used in the terms of service
- Data Security important so competitors don’t get sales data

MVP development and lessons learned (how did this impact your architecture choices?)

Streamlit (Hugging Face):

- Good for simple or small scale app development but implementation with a Flask app or other framework allows for more flexibility
- Hugging Face is good for community involvement or small projects but the lack of DVC operability makes it unrealistic when scaling. EC2 or Heroku may be preferable when incorporating user provided data

MVP:

- Not intended to be a perfect production version, but needs all of the vital features especially to enable actual use and feedback/data to further improve the product

MVP demo

17691 Project MVP - a Hugging Face Space by ewclark47